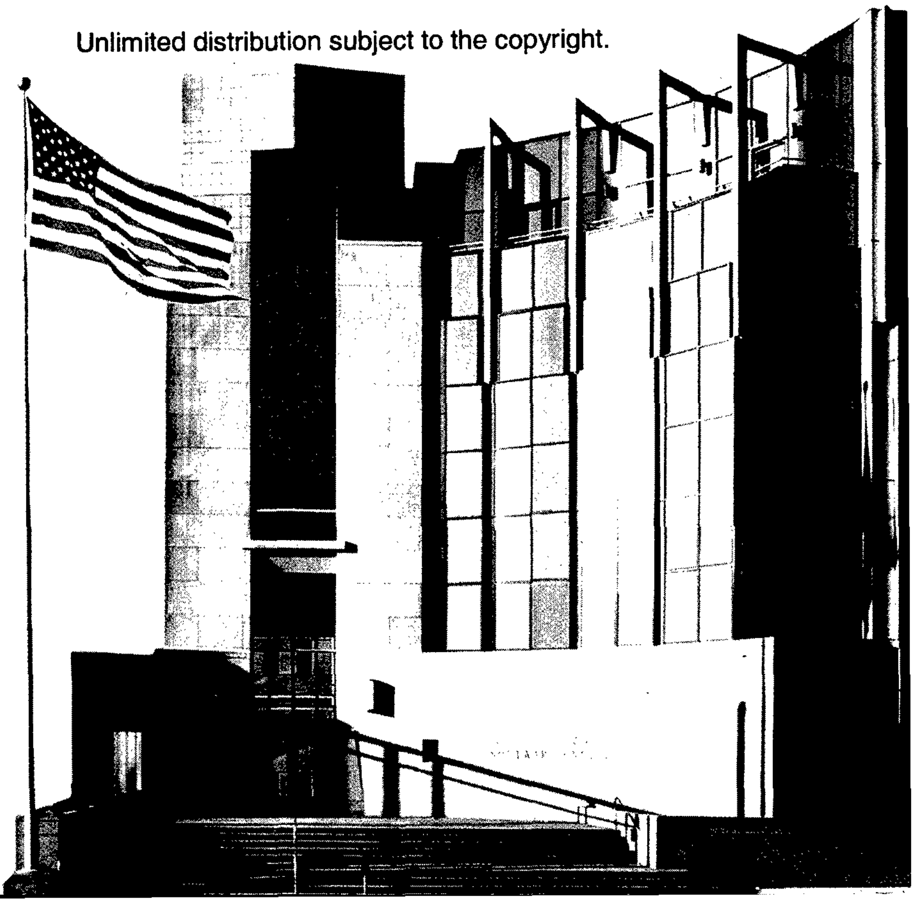
Carnegie Mellon
Software Engineering Institute

Software Architecture in DoD Acquisition: A Reference Standard for a Software Architecture Document

John K. Bergey
Paul C. Clements

February 2005

Software Architecture Technology Initiative



Technical Note
CMU/SEI-2005-TN-020

Software Architecture in DoD Acquisition: A Reference Standard for a Software Architecture Document

John K. Bergey
Paul C. Clements

February 2005

Software Architecture Technology Initiative

Technical Note
CMU/SEI-2005-TN-020

Contents

Acknowledgments

We would like to thank Lawrence Jones, Linda Northrop, and Reed Little for their careful reviews.

About the Technical Note Series on Software Architecture Practices in the Department of Defense

The Product Line Systems Program at the Carnegie Mellon® Software Engineering Institute (SEI) is publishing a series of technical notes designed to condense knowledge about software architecture practices into a concise and usable form for the Department of Defense (DoD) acquisition manager and practitioner. Our objective is to provide practical guidance and lay a conceptual foundation for DoD architecture practice. This series, called Software Architecture Practices in the Department of Defense, is a companion to the SEI series on product line acquisition and business practices [Campbell 02, Bergey 01a, Cohen 01, Bergey 00a, Bergey 00b, Jones 99, Bergey 99].

This technical note is part of a special series of reports titled "Software Architecture in DoD Acquisition" that is aimed at DoD acquisition specialists who are commissioning large software-intensive systems for the DoD. The intent of the series is to explain how to bring the concepts of software architecture effectively into the system acquisition process. Titles currently in the series include

- *A Reference Standard for a Software Architecture Document*: This technical note suggests the layout and contents of each section of a Software Architecture Document. [This report is the one you are now reading.]
- *An Approach and Language for a Software Development Plan*: This technical note offers an example approach and corresponding language that covers software architecture practices and that could be inserted into a contractor's software development plan (SDP) [Bergey 05].

Possible future titles include

- *Reviewing a Software Architecture Document*: This technical note will provide a step-by-step approach for the peer review of software architecture documentation, including specific questions and a methodological basis for achieving high-quality reviews.
- *An Approach and Language for a Software Architecture Evaluation Plan*: This technical note will offer an example approach and corresponding language for creating a plan to conduct a series of in situ software architecture evaluations in a system acquisition using the SEI Architecture Tradeoff Analysis Method® (ATAM®).

® Carnegie Mellon, Architecture Tradeoff Analysis Method, and ATAM are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

- *A Reference Standard for a Software Architecture Evaluation Report*: This technical note will suggest the layout and contents of each section of an ATAM evaluation report.

Abstract

The right software architecture is essential for a software-intensive system to meet its functional requirements as well as its quality requirements that govern real-time performance, reliability, maintainability, and a host of other quality attributes. Because an architecture comprises the earliest, most important, and most far-reaching design decisions, it is important for an acquisition organization to exercise its oversight prerogatives with respect to software architecture. Having the right software architecture documentation is a prerequisite for managing and guiding a software development effort and conducting in situ software architecture evaluations. Conducting an architecture evaluation to determine the software architecture's fitness for purpose is one of the most powerful, technical risk mitigation strategies available to a program office.

This report provides an example reference standard for a Software Architecture Document (SAD). An acquisition organization can use this standard to contractually acquire the documentation needed for communicating the software architecture design and conducting software architecture evaluations. The example used in this report is drawn from an actual SAD written by a major U.S. Department of Defense contractor in a weapon system acquisition. The intent of this report is to provide an example for other acquisition efforts to use (and adapt as appropriate) in their own procurements.

1 Introduction

The right software architecture is essential for a software-intensive system to meet its behavioral (functional) requirements as well as its quality requirements that govern real-time performance, security, reliability, maintainability, and a host of other quality attributes. Software architecture is especially critical in large, complex systems because software is a major contributor to the cost and quality of such systems and to the schedule and risk of acquiring them.

Documenting the architecture effectively is as important as crafting it because the documentation is how the architecture is communicated to its many stakeholders. An architecture that is not communicated clearly, completely, and unambiguously to developers, downstream designers, testers, analysts, certification authorities, and other key practitioners is nearly useless and will require a multitude of architects to constantly and repeatedly explain themselves. In a very short time—let alone over a long-lived system's expected operational lifetime—the architecture will lose all integrity.

Assuring that the architecture is being documented effectively is difficult enough in the context of in-house development, but an acquisition organization has an even harder task because its contact and leverage points with the contractor(s) are limited, occur at discrete points in the life cycle, and are exercised from a distance.

Nevertheless, it is important for an acquisition organization that commissions the development of large software systems to exercise its oversight prerogatives with respect to software architecture. Such an organization is the U.S. Department of Defense (DoD). Because an architecture comprises the earliest, most important, and most far-reaching design decisions, conducting an architecture evaluation to determine the software architecture's fitness for purpose is one of the most powerful, technical, risk mitigation strategies available to a DoD program office. Such an evaluation is impossible without adequate architecture documentation.

Bergey and colleagues describe the contracting mechanisms that can enable a DoD organization to integrate software architecture evaluations into its system acquisition strategy effectively and in a manner that will serve the program throughout its life cycle [Bergey 00c, Bergey 01b].

This report covers another avenue of exercising architectural control—the Software Architecture Document (SAD)—and provides a standard for it. Suitable software architecture documentation is needed not only to guide and manage the software development effort, but also as a prerequisite for conducting a software architecture evaluation using a method such as the Architecture Tradeoff and Analysis Method® (ATAM®) developed by the Software Engineering Institute [Clements 01]. Establishing the expected content and organization of an SAD helps to insure that the proper information is captured in only one place. Using a standard document organization

- organizes the information so the reader can navigate the document and find specific information quickly. Software documentation might be read from cover to cover at most once, probably never. But an SAD is likely to be referenced hundreds or thousands of times.
- helps the document writer plan and organize the content and identify any unfinished work (such work appears under sections labeled "TBD" for to be determined)
- embodies completeness rules for the information; the sections of the document constitute the set of important aspects that need to be conveyed. Hence, the standard organization can form the basis for a first-order validation check of the document at review time.

The purpose of this report is to provide an example description of an SAD that is suitable for communicating the software architecture design and conducting in situ software architecture evaluations.

The SAD outline in this report is based on the prescriptive advice for architecture documentation from Clements and colleagues [Clements 02]. The outline is also consistent with the best practice for architecture documentation for software-intensive systems recommended by the Institute of Electrical and Electronics Engineers, Inc. (IEEE) [IEEE 00]. While this SAD organization has been used in practice, it remains an example; for this reason, you may need to modify it to suit the specific needs of your particular project or acquisition. Before incorporating the outline, you should understand the purpose and relevance of each section with regard to your organization.

1.1 Producing the SAD in Multiple Volumes

One common modification is to split the SAD into multiple volumes. Doing so often helps make the documentation more accessible and usable. You can split the SAD in many ways, but keep in mind that its structure must support the needs of the intended audience and be determined in the context of the project. Each document that you produce should include the date of issue, status (e.g., draft or baseline), version number, name of issuing organization, change history, and a summary. A few decomposition options are

- **A Two-Volume approach:** Separate the documentation into two volumes; one containing the specific views of the software architecture and one containing everything else.

® Architecture Tradeoff Analysis Method and ATAM are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

- **A Three-Volume approach:** Document organizational policies, procedures, and the directory in one volume, system-specific overview material in a second, and view documentation in a third.
- **A Four-Volume approach:** Create one volume for each viewtype defined by Clements and colleagues: module, component-and-connector, and allocation [Clements 02] and include the documentation for the relevant views. Then, include all the other information in the fourth volume.

1.2 Mechanism for Requiring the Desired Content

The reference standard for SAD contents has not yet been made an official data item description (DID) that can be called out in an RFP. However, another mechanism to incorporate desired information into a document is the Contract Data Requirements List (CDRL). The CDRL lists all the deliverables on the contract and specifies things such as when the contract is to be delivered and how many copies are required. A common approach is to call out a standard DID in Block 1 of the CDRL, such as for a Software Description Document (SDD) and then, in Block 3, to assign a subtitle, such as "Software Architecture Document." In the "Remarks" section, instructions for tailoring the referenced DID are allowed. For example, you might make the following customizations:

> Replace Sections 6.12.3.c and 6.12.3.d [which are specific sections in the original SDD DID] with the following text:
>
> ...where "the following" can be the reference standard given in the following section.

2 Software Architecture Document (SAD) Reference Standard

The SAD reference standard is shown below.

General Instructions for Using This Standard

- *Alternate presentation styles.* Diagrams, tables, matrices, and other presentation styles are acceptable substitutes for text when they make data required by this reference standard more readable. However, all diagrams must include a key that (a) explains the meaning of each symbol and connecting line used in the diagram, (b) cites another document that contains such information using a complete citation that enables the reader to easily locate the referenced document, or (c) names a well-known standard notation along with its version number (e.g., UML V1.4).
- *Title page.* Include a title page containing the following items, as applicable: the document number; the volume number; a version/revision indicator; security markings or other restrictions on the handling of the document; the publication date; the document title; the name, abbreviation, and any other identifier for the system, subsystem, or item to which the document applies; the contract number; the name and address of the preparing organization; the applicable distribution statement; and the date of approval.
- *Table of contents.* Include a table of contents providing the number, title, and page number of each titled paragraph, figure, table, and appendix.
- *Page numbering/labeling.* Include the document number and page number on each page.
- *Substitution of and reference to existing documents.* References, including hyperlinks to their locations in the project's online document repository, should be substituted for all or part of the document if they contain the appropriate data and are under project configuration control.
- *Tailoring.* The contractor's format and tailoring are permitted with a waiver.
- *Delivery and format.* Deliver data on PC CD-ROM, through secure email, or via an integrated digital environment. Use MS Office 2000 or a later release for draft and final versions of the SAD. Additionally, provide a final copy as an Adobe PDF file. The contractor's format is acceptable.

Terms Used in This Standard

- *Document* means a collection of data regardless of its medium or the number of volumes it consists of. Use of electronic data is encouraged.
- A *viewpoint* is a pattern or template that specifies the conventions for constructing and using a view. It is created after three things have been determined: (1) the view's purpose, (2) the view's audience, and (3) how the view will be created and analyzed [IEEE 00]. Once created, viewpoints are used to develop multiple individual views.
- A *view* is a representation of a whole system from the perspective of a related set of concerns [IEEE 00]. A view is described by the types of elements and relations that it contains, as well as any constraints on their interaction. A view conforms to a viewpoint.
- A *view packet* specifies a portion of the system in a view. It is the smallest bundle of useful documentation that can be given to a stakeholder [Clements 02]. View packets usually show large areas of the system at shallow depth or small areas of the system at great depth.

Content Requirements

1. Documentation Roadmap and Overview
In this section, provide information that will help readers of the SAD find the information they need quickly. Structure the information in the sections listed below.

1.1 Document Management and Configuration Control Information
Identify the version, release date, and other relevant management and configuration control information associated with the current version of the document. Optional: Include a change history, highlighting significant changes from version to version.

1.2 Purpose and Scope of the SAD
Explain the SAD's overall purpose and scope. Explain the criteria for deciding which design decisions are architectural (and therefore documented in the SAD) and which are non-architectural (and therefore documented elsewhere).

1.3 How the SAD Is Organized
Provide a narrative description and overall contents of the seven major sections of the SAD (as identified by this reference standard).

1.4 Stakeholder Representation

1.4.1 Stakeholders and Their Concerns
Provide a list of the stakeholder roles considered in the development of the architecture described by this SAD. For each role, list the stakeholder concerns that can be addressed by the information in this SAD. A convenient way to represent this information is as a matrix, where the rows list stakeholder roles, the columns list concerns, and the cells indicate how serious the concern is to a stakeholder in that role. The following stakeholders shall be considered at a minimum:

- application software developers
- infrastructure software developers
- end users
- project segment teams

- application system engineers
- application and platform hardware engineers
- security engineers and certifiers
- safety engineers and certifiers
- communications engineers
- system-of-system engineers
- chief engineer/chief scientist
- lead system integrator (LSI) program management
- government program management (including those concerned with licensing)
- system integration and test engineers
- external test agencies
- operational system managers
- trainers
- maintainers
- other service representatives
- auditors (LSI internal, the Government Accounting Office, etc.)
- representatives of standardization activities

1.4.2 Stakeholder Scenarios for Using the SAD

For each role identified in Section 1.4.1, write a few short scenarios that explain how stakeholders in that role would use specific sections of the SAD to help address their concerns.

1.5 Viewpoint and View Definitions

Introduction to Viewpoints. Define the term *viewpoint* and describe how it's used in this SAD.

Describe each viewpoint used in the SAD as outlined in Section 1.5.i. The following viewpoints must be included:

- communications viewpoint. Views conforming to this viewpoint show the communication paths used by software-initiated or software-carried messages and the software elements that send and receive information along those paths. Views conforming to this viewpoint provide the basis for analysis for determining whether necessary communication bandwidth and performance will be achieved, thus enabling the system to meet its operational requirements that depend on communication.
- data load viewpoint. Views conforming to this viewpoint show the data required and provided by software elements and show where that data is stored and how it is backed up and recovered in the event of loss. Views conforming to this viewpoint provide the basis for analysis for determining whether units will have access to the necessary information.
- information assurance viewpoint. Views conforming to this viewpoint show the location and flow of classified or otherwise sensitive information in the system, as well as elements that provide essential services that must be protected from denial-of-service attacks. Views conforming to this viewpoint provide the basis for analysis for determining whether the system's information-assurance needs will be met.
- safety viewpoint. Views conforming to this viewpoint show elements that provide safety-critical functionality and how they are used. Views

conforming to this viewpoint provide the basis for analysis for determining whether the system's safety needs will be satisfied.

- reliability viewpoint. Views conforming to this viewpoint show elements that provide mission-critical functionality, how they are used, and any redundancy or failover capabilities provided to assume that functionality in the event of failure. Views conforming to this viewpoint provide the basis for analysis for determining whether the system's reliability needs will be satisfied.

1.5.i Viewpoint #i

Name the viewpoint.

1.5.i.1 Abstract

Provide a brief overview of the viewpoint.

1.5.i.2 Stakeholders and Their Concerns to Be Addressed

Describe the stakeholders and their concerns that this viewpoint is intended to address. List questions that can be answered by consulting views that conform to this viewpoint. Optionally, include significant questions that cannot be answered by consulting views conforming to this viewpoint.

1.5.i.3 Elements, Relations, Properties, and Constraints

Define the types of elements, the relations among them, the significant properties they exhibit, and the constraints they obey for views conforming to this viewpoint.

1.5.i.4 Language(s) to Model/Represent Conforming Views

List each language that will be used to model or represent views conforming to this viewpoint and cite a definition document for it.

1.5.i.5 Applicable Evaluation/Analysis Techniques and Consistency/Completeness Criteria

1.5.i.6 Viewpoint Source

Provide a citation for the source of this viewpoint definition, if any.

1.6 How a View Is Documented

Describe the documentation organization for documenting a view.

1.7 Relationship to Other SADs

Describe the relationship between this SAD and other architecture documents, both system and software.

1.8 Process for Updating This SAD

Describe the process a reader should follow to report discrepancies, errors, inconsistencies, or omissions from this SAD. Include necessary contact information for submitting such a report. If a form is required, either include a copy of the blank form that can be photocopied or refer to an online electronic version. Describe how error reports are handled and how and when a submitter will be notified of the issue's disposition.

2. Architecture Background

2.1 Problem Background

In this section, explain the constraints that significantly influenced the architecture. Structure the information in the following sections:

2.1.1 System Overview

Describe the general function and purpose of the system or subsystem whose architecture is described in this SAD. If appropriate, include a context diagram showing the system or subsystem, and other systems or subsystems with which it communicates or interoperates.

2.1.2 Goals and Context

Describe the goals and major contextual factors for the software architecture. Include a description of the role software architecture plays in the life cycle, relevant acquisition factors, the impact of the LSI model, the effects of incremental development, and the relationship to system engineering results and artifacts.

2.1.3 Significant Driving Requirements

Describe behavioral and quality attribute requirements (original or derived) that shaped the software architecture. Include any scenarios that express driving behavioral and quality attribute goals, such as those crafted during an ATAM evaluation [Clements 01].

2.2 Solution Background

In this section, provide a description of why the architecture is the way it is and why it is appropriate for satisfying the functional and quality attribute goals levied upon it. Structure the information in the following sections:

2.2.1 Architectural Approaches

Provide a rationale for the major design decisions embodied by the software architecture. Describe any design approaches applied to the software architecture—including the use of architectural styles or design patterns—when the scope of those approaches transcends any single architectural view. Explain why those approaches were chosen and specifically why they were chosen over other seriously considered approaches. Describe any relevant issues dealing with commercial off-the-shelf (COTS) or government off-the-shelf (GOTS) components, including any associated trade spaces.

2.2.2 Analysis Results

Describe the results of any quantitative or qualitative analyses that have proven the software architecture is fit for purpose. If an architecture evaluation has been performed using the ATAM or a comparable method, include the analysis sections of the final evaluation report. Refer to the results of any other relevant trade studies, quantitative modeling, or other analysis results.

2.2.3 Requirements Coverage

Describe the requirements (original or derived) addressed by the software architecture. Include those requirements or constraints that are derived from higher level SADs.

2.2.4 Summary of Changes in Current Version

For versions of the SAD after the original release, summarize the actions; the decisions and decision drivers; the requirements changes and analysis and trade study results that became decision drivers; and explain how these decisions caused the architecture to evolve or change.

2.3 Product Line Reuse Considerations

When a product line is being developed, this section details how the software covered by this SAD is planned or expected to be reused in order to support the product line vision. In particular, this section includes a complete list of the variations that are planned to be produced and supported. *Variation* refers to a variant of the software produced through the use of preplanned variation mechanisms made available in the software architecture. Variation may refer to a variant of a module or a collection of modules identified in this SAD, or to the entire system or subsystem covered by this SAD. For each variation, this section identifies the increment(s) of the software build in which the variation will be available and used. Finally, this section describes any additional potential that exists to reuse one or more of the modules or their identified variations, even if this reuse is not currently planned for any increment.

3. Views

Describe each view using the outline below. The SAD should contain one view for each viewpoint listed in Section 1.5.

3.i View # i

Name the view.

3.i.1 View Description

Describe the view's purpose and contents. Refer to the viewpoint description in Section 1.5 to which this view conforms.

3.i.2 View Packet Overview

Show the set of view packets in this view and explain why the set is complete and non-duplicative.

3.i.3 Architecture Background

Provide any architecture background (including significant driving requirements, design approaches, patterns, analysis results, and requirements coverage) that applies to this view.

3.i.4 Variability Mechanisms

Describe any architectural variability mechanisms (e.g., adaptation data, compile-time parameters, and variable replication) described by this view, including a description of how and when those mechanisms can be exercised and any constraints on their use.

3.i.5 View Packets

Describe each view packet in the view using the following outline:

3.5.i.j View Packet # j

Name the view packet.

3.5.i.j.1 Primary Presentation

Using an appropriate language, languages, notation, or tool-based representation, present the elements that populate this view packet and the relations among them.

3.5.i.j.2 Element Catalog

3.5.i.j.2.1 Elements

Describe each element shown in the primary presentation, along with the values of its relevant properties, which are described in the viewpoint to which this view conforms.

3.5.i.j.2.2 Relations

Describe any additional relations among elements shown in the primary presentation and any specializations or restrictions on those relations.

3.5.i.j.2.3 Interfaces

Specify the software interfaces to any elements shown in the primary presentation that must be visible to other elements.

3.5.i.j.2.4 Behavior
Specify any significant behavior of elements or groups of interacting elements that are shown in the primary presentation.

3.5.i.j.2.5 Constraints
List any constraints on elements or relations not otherwise described.

3.5.i.j.3 Context Diagram
Provide a context diagram showing the context of the part of the system represented by this view packet. Designate the view packet's scope with a distinguished symbol and show interactions with external entities in the view's vocabulary.

3.5.i.j.4 Variability Mechanisms
Describe any variabilities that are available in the portion of the system shown in the view packet, along with how and when those mechanisms can be used.

3.5.i.j.5 Architecture Background
Provide the rationale for any significant design decisions whose scope is limited to this view packet.

3.5.i.j.6 Related View Packets
Provide section references for related view packets, including the parent, children, and siblings of this view packet. Related view packets might be in the same view or in different ones.

4. Relations Among Views

4.1 General Relations Among Views
Describe the general relationship among the views chosen to represent the architecture. Discuss the consistencies among those views and identify any known inconsistencies.

4.2 View-to-View Relations
Show how the elements in related views are associated.

5. Referenced Materials
For each referenced document, provide a complete citation that enables a reader to easily locate the document.

6. Directory
Provide an index of all element, relation, and property names. For each one, identify where in the SAD it is defined and used.

6.1 Index
Provide an index of all element, relation, and property names. For each one, identify where in the SAD it was defined and used.

6.2 Glossary
Provide definitions of the special terms used in the SAD. If a term has a different meaning in this SAD than it does in a parent SAD, explain why.

6.3 Acronym List

Provide definitions of the acronyms used in the SAD.

A. Appendixes

Appendixes can be used to provide information published separately for convenience in document maintenance (e.g., charts, classified data). As applicable, each appendix should be referenced in the main body of the document where the appendix data would normally have been provided. Appendixes can be bound as separate documents for ease in handling.

References

URLs are valid as of the publication date of this document.

[Bergey 99] Bergey, J.; Fisher, M.; & Jones, L. *The DoD Acquisition Environment and Software Product Lines* (CMU/SEI-99-TN-004, ADA244787). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1999. http://www.sei.cmu.edu/publications/documents/99.reports /99tn004/99tn004abstract.html

[Bergey 00a] Bergey, J. & Smith, D. *Guidelines for Using OAR Concepts in a DoD Product Line Acquisition Environment* (CMU/SEI-2000-TN-004, ADA377385). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000. http://www.sei.cmu.edu/publications/documents/00.reports /00tn004.html

[Bergey 00b] Bergey, J.; Fisher, M.; Gallagher, B.; Jones, L.; & Northrop, L. *Basic Concepts of Product Line Practice for the DoD* (CMU/SEI-2000-TN-001, ADA375859). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000. http://www.sei.cmu.edu/publications/documents/00.reports /00tn001.html

[Bergey 00c] Bergey, J. & Fisher, M. *Software Architecture Evaluation with ATAM in the DoD System Acquisition Context* (CMU/SEI-99-TN-012, ADA377450). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000. http://www.sei.cmu.edu/publications/documents/99.reports /99tn012/99tn012abstract.html

[Bergey 01a] Bergey, J. & Goethert, W. *Developing a Product Line Acquisition Strategy for a DoD Organization: A Case Study* (CMU/SEI-2001-TN-021, ADA395202). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2001. http://www.sei.cmu.edu/publications/documents/01.reports /01tn021.html

[Bergey 01b]	Bergey, J. & Fisher, M. *Use of the Architecture Tradeoff Analysis Method (ATAM) in the Acquisition of Software-Intensive Systems* (CMU/SEI-2001-TN-009, ADA396096). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2001. http://www.sei.cmu.edu/publications/documents/01.reports /01tn009.html
[Bergey 05]	Bergey, J. & Clements, P. *Software Architecture in Acquisition: An Approach and Language for a Software Development Plan* (CMU/SEI-2005-TN-019). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2005. http://www.sei.cmu.edu/publications/documents/05.reports /05tn019.html
[Campbell 02]	Campbell, G. *A Software Product Line Vision for Defense Acquisition* (CMU/SEI-2002-TN-002, ADA403810). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002. http://www.sei.cmu.edu/publications/documents/02.reports /02tn002.html
[Clements 01]	Clements, P.; Kazman, R.; & Klein, M. *Evaluating Software Architectures: Methods and Case Studies*. Boston, MA: Addison-Wesley, 2001.
[Clements 02]	Clements, P.; Bachmann, F.; Bass, L.; Garlan, D.; Ivers, J.; Little, R.; Nord, R.; & Stafford, J. *Documenting Software Architectures: Views and Beyond*. Boston, MA: Addison-Wesley, 2002.
[Cohen 01]	Cohen, S. *Case Study: Building and Communicating a Business Case for a DoD Product Line* (CMU/SEI-2001-TN-020, ADA395155). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2001. http://www.sei.cmu.edu/publications/documents/01.reports /01tn020.html
[IEEE 00]	Institute of Electrical and Electronics Engineers. *Recommended Practice for Architectural Description of Software-Intensive Systems* (IEEE Std 1471-2000). New York, NY: Institute of Electrical and Electronics Engineers, 2000.

[Jones 99] Jones, L. *Product Line Acquisition in the DoD: The Promise, The Challenges* (CMU/SEI-99-TN-011, ADA373184). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1999. http://www.sei.cmu.edu/publications/documents/99.reports/99tn011/99tn011abstract.html

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE February 2005	3. REPORT TYPE AND DATES COVERED Final
4. TITLE AND SUBTITLE Software Architecture in DoD Acquisition: A Reference Standard for a Software Architecture Document		5. FUNDING NUMBERS F19628-00-C-0003
6. AUTHOR(S) John K. Bergey, Paul C. Clements		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213		8. PERFORMING ORGANIZATION REPORT NUMBER CMU/SEI-2005-TN-020
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116		10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES		
12A DISTRIBUTION/AVAILABILITY STATEMENT Unclassified/Unlimited, DTIC, NTIS		12B DISTRIBUTION CODE

13. ABSTRACT (MAXIMUM 200 WORDS)

The right software architecture is essential for a software-intensive system to meet its functional requirements as well as its quality requirements that govern real-time performance, reliability, maintainability, and a host of other quality attributes. Because an architecture comprises the earliest, most important, and most far-reaching design decisions, it is important for an acquisition organization to exercise its oversight prerogatives with respect to software architecture. Having the right software architecture documentation is a prerequisite for managing and guiding a software development effort and conducting in situ software architecture evaluations. Conducting an architecture evaluation to determine the software architecture's fitness for purpose is one of the most powerful, technical, risk mitigation strategies available to a program office. This report provides an example reference standard for a Software Architecture Document (SAD). An acquisition organization can use this standard to contractually acquire the documentation needed for communicating the software architecture design and conducting software architecture evaluations. The example used in this report is drawn from an actual SAD written by a major U.S. Department of Defense contractor in a weapon system acquisition. The intent of this report is to provide an example for other acquisition efforts to use (and adapt as appropriate) in their own procurements.

14. SUBJECT TERMS software architecture, architecture documentation, architecture evaluation, architecture in acquisition, software development plan, software architecture data item description, software architecture DID			15. NUMBER OF PAGES 26
16. PRICE CODE			
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL